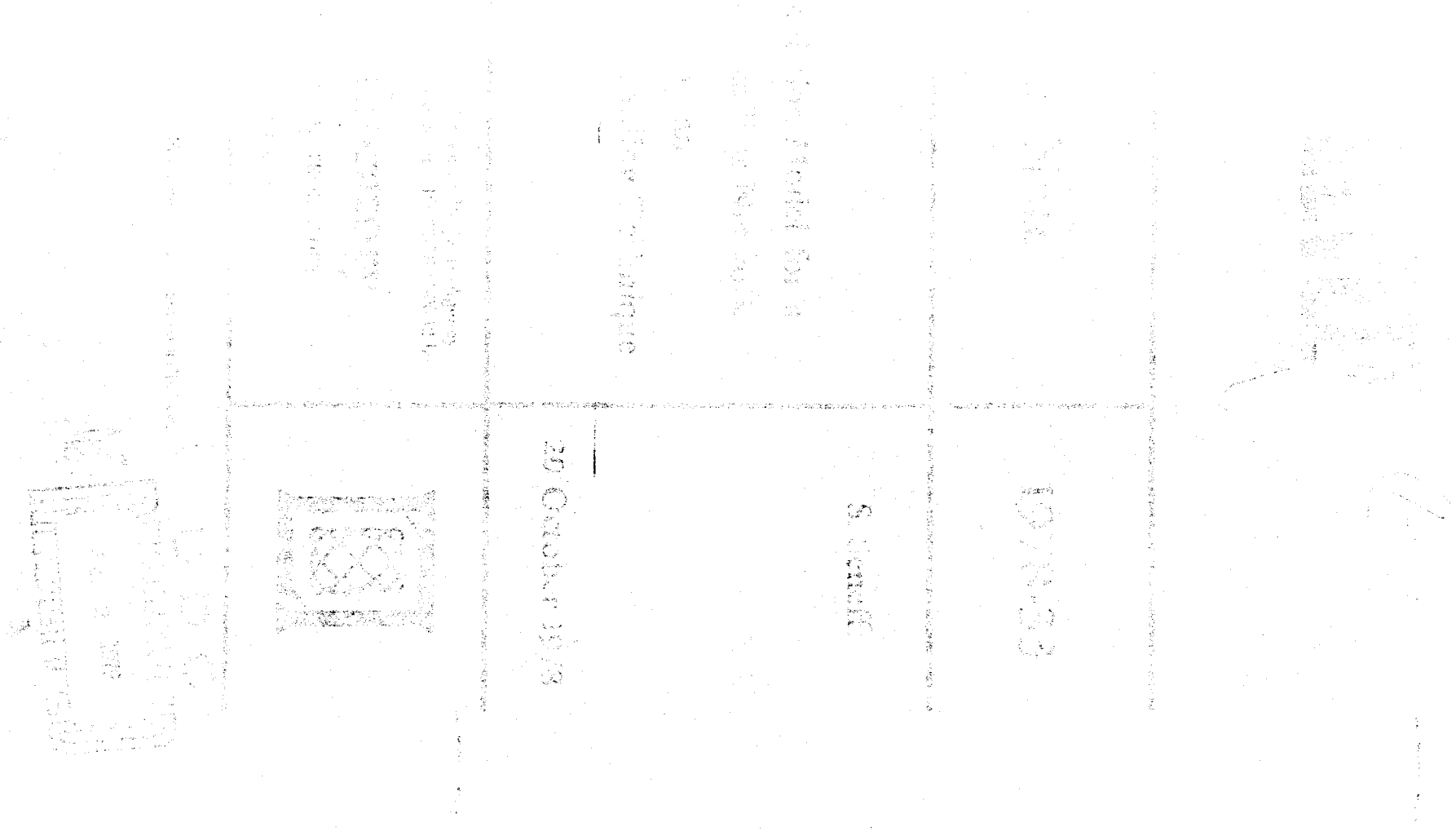
AD A065182

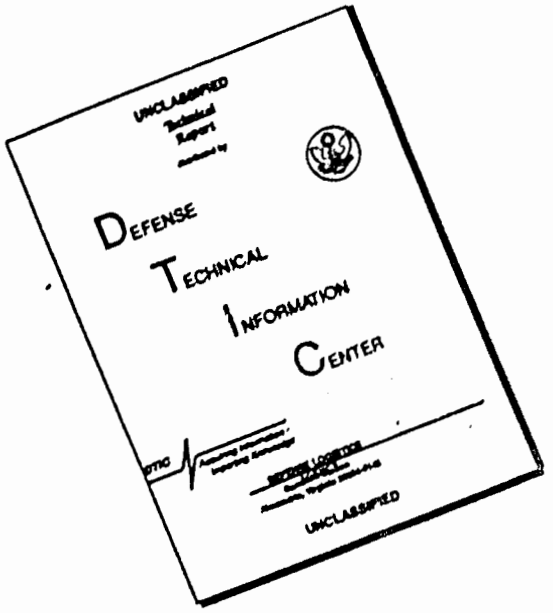


...search documented in this document was performed at Lincoln Laboratory, ... operated by Massachusetts Institute of Technology. ... sponsored by the Department of the Air Force under Contract F19628-95-C-0002 (ARPA Order 200).

...report has been reviewed and is approved for publication.

...

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

LINCOLN LABORATORY

COMPUTER SIMULATION MODEL FOR A DIGITAL COMMUNICATIONS NETWORK UTILIZING AN EMBEDDED SPEECH ENCODING TECHNIQUE

S. SENEFF

Group 24

TECHNICAL NOTE 1978-33

20 OCTOBER 1978

LEXINGTON MASSACHUSETTS

ABSTRACT

This note describes a computer simulation of a model of a rate-adaptive packetized network, based on the assumption that users have available to them an embedded-coding type vocoder. Queues are controlled at each link by stripping off lower priority packets of the encoded speech. Three issues addressed in the simulation were:

1) The development of an adequate packet stripping strategy at each node,

2) The development of an end-to-end feedback strategy that would give stable results and yield an overall higher performance than was realized in the absence of feedback, and

3) The integration of the speech packets with data packets.

Conclusions are that a judiciously chosen end-to-end feedback strategy can not only improve the efficiency of the system but also alleviate the problem of sudden rate drops due to link overload. The inclusion of data traffic in the model resulted in overall higher link utilization than was realized in the absence of data.

CONTENTS

1. INTRODUCTION

Due to the recent revolution in digital technology, it can be confidently predicted that in the near future digital voice terminals will be available at a sufficiently low cost and compact size to make feasible the possibility of large scale digital transmission of the speech signal [1]. Given this possibility, it has become important to address the issues involved in the efficient transmission of speech over a complex communications network. Recent efforts include attempts to incorporate voice traffic into an already existing packetized data network (the ARPAnet) [2], and computer simulation experiments designed to determine efficient design and control strategies for voice/data integration [3-7]. One issue is the question of whether voice should be transmitted over a packetized or a circuit switched network. Forgie and Nemeth [5] have proposed a "packetized virtual circuit" (PVC) strategy which attempts to take advantage of the features of both types of systems. With such a system, packet header information is kept at a minimum, thus allowing for relatively efficient transmission of the short speech packets that are preferred for reduced delay times. Because the packetized concept is retained, the speech bit rate can be variable (and, in particular, transmission can cease during silence intervals), and speech and data packets can be naturally integrated to share the same network communications channels.

A major requirement for successful transmission of speech across a digital network is that delays at each node be kept under a small fraction of a second. This requirement is in direct conflict with the fact that bit flow typically fluctuates significantly over short time intervals. Such fluctuations could ordinarily be smoothed out at each link by allowing the queue to expand and shrink as needed. However, with speech, the large variations in the delay imposed by the resulting fluctuating queue will create problems at the synthesizer, which requires a continuous steady flow of bits to avoid glitches or annoying time delays in the conversation.

Clearly, one source of variation is simply that the number of users will vary with time. Connections will continually be made and broken as conversations are initiated and completed. A much more rapid fluctuation will occur if users make use of TASI (Time-Assigned Speech Interpolation) [8] to reduce the mean data flow for a given vocoder rate. If the users only transmit during talkspurt intervals, a greater than 50% decrease in the mean data flow for a given vocoder bit rate could be realized. However, this is realized at the expense of a much more rapid variation in the actual bit flow, since users go into and out of talkspurt mode much more frequently than the rate at which new users enter and leave the system. If the network is able to handle both data and speech, then there are even greater fluctuations in the load, due to

variations in the data traffic. And finally, the link capacity itself may vary over time, due to fading or jamming, further necessitating some kind of control strategy for regulating traffic flow.

Unless there is some controlled means of reducing or increasing the bit rate per customer at will, based on the instantaneous customer load at a link, there is no other alternative, given the above variables, than to restrict the number of users on a given link such that on the average the utilization of the link is far below capacity. One could envision a variety of possible control strategies. The simplest such strategy would be a negotiation with the network for a vocoder bit rate at the initiation of each call. The appropriate vocoder would be loaded into both the transmitter and the receiver, and the bit rate for that conversation would be fixed until termination. Such a technique would probably be too sluggish to deal adequately with the variations in the load, since conversations tend to be of several minute durations.

A second, far more cumbersome, alternative would be for each link in the path to communicate back to transmitters anticipated conditions of overload. Transmitters would respond by replacing the vocoder with a new one, of lower rate. There are several problems inherent in this method. First is the complex bookkeeping

necessary for separate link-to-transmitter paths from all links to all transmitters. Second is the critical timing problem necessary for the receiver to load in the new vocoder in time to interpret properly the bits now arriving with an entirely different encoding structure. Third is the fact that the loading process will take a significant amount of time during which speech cannot be vocoded, and glitches will undoubtedly occur when rates are changed. Finally, there is a critical time delay between when the link is aware of imminent catastrophe and when the vocoder can act on such awareness. The response to overload conditions may not be rapid enough to avoid crisis.

With either of the above two methods, in the event that the link becomes overloaded in spite of the (predicted to be inadequate) control strategies, the only options open to the link are to discard bits as queues build up to extreme levels. Such discards will introduce gaps in the synthesized speech.

The third and most enticing method of controlling rate per customer gives instantaneous response, requires no special paths from links to transmitters, and avoids gaps in the speech due to either reloading of machines or the discard of bits. This method, first proposed by the Naval Research laboratory, assumes that users have available to them a specialized "embedded-coding vocoder": a vocoder whose analyzer produces bits at a high rate, but whose

synthesizer can reconstruct speech at a variety of different rates, from selected subsets of the bits produced by the analyzer. A preliminary version of such a vocoder has recently been developed at Lincoln Laboratory [9], and there is an ongoing research effort to improve its performance. Some of the issues involved for a strategy for adaptive rate control, given such a vocoder, are addressed in [10]. This report describes a simulation experiment designed to develop a further understanding of the performance of such a vocoder in a network environment.

An embedded-coding vocoder fits quite naturally into a packetized, as opposed to a circuit switched, network. As shown in Figure 1, the analyzer would produce a set of packets with distinct priority labels. Packets of priority one contain the information necessary to reconstruct speech at the lowest available bit rate (in the example, 2400 bits/sec). Bits in the packets of priority level two can be added to the bits in priority level one packets to produce speech at the next higher bit rate (4800 bits/sec). The synthesizer can reconstruct the highest quality or most robust speech if packets of all priorities have arrived. With any intermediate number of packets, there will be an intermediate reconstruction.

Given such a vocoder, the network task is much simpler. Each link in the network controls its incoming bit rate by discarding

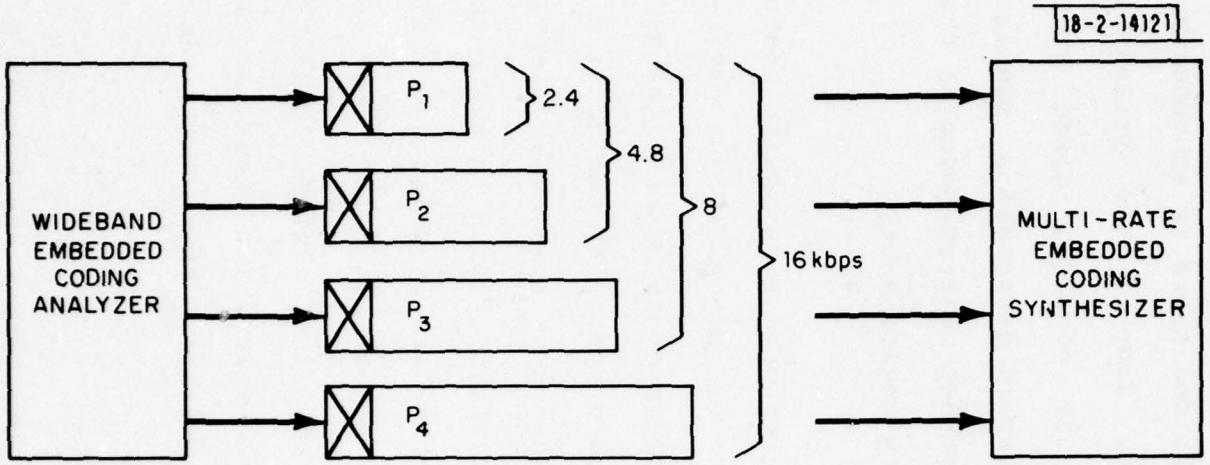


Fig. 1. Example of an embedded-coding vocoder capable of synthesizing at the rates 2400, 4800, 8000, and 1600 bps. Analyzer transmits packets at four priorities, and network may strip all but priority one packets.

packets of low priority. At a given instant in time, the link has a lowest-acceptance-priority or stripping-priority level. Packets of lower priority arriving at that link are discarded. By constantly monitoring either the queue or the incoming bit rate, the link can exercise tight control of overloads or underloads by simply reassigning its stripping-priority level.

The problem becomes much more interesting if one assumes the possibility of improving the overall performance by introducing a strategy of end-to-end feedback. Without feedback, the analyzer always transmits bits of all priority levels; in our example, 16,000 bits/sec. Bits are discarded at various points in the network, and the receiver finally receives a subset of the total. Any bits discarded at nodes other than the first node in the path are bits which needlessly jammed all preceding links in the path. In particular, if the last link in the path could accept only priority one packets, (due for example to a large data file transfer), all of the other links in the path would have been needlessly overloaded with bits which had to be discarded just short of their destination.

If, instead, the receiver had informed the transmitter that he was only receiving priority one packets, the transmitter could respond by only transmitting priority one packets, and the load on intermediate links in the path would be relieved. While the result

would not improve the situation for the particular conversation utilizing the feedback, it would possibly allow other users, sharing intermediate nodes, to operate at a higher rate, because bits formerly discarded just short of their destination are now being discarded at the source.

If feedback is to exist, then there must also be probing [10], in order for users to be able to recover from an easing up of the network load. Continuing with the example above, after the data-file transfer at the last link has been completed, the link can now accept a much higher bit rate than before. However, the synthesizer, in the case of feedback, will continue to report that only priority one packets are being received, since these are all that are being sent. The transmitter will have to be allowed some strategy for periodically increasing his rate for a temporary period, and then sustaining the increase if the receiver reports a successful transmission of the lower priority packets. An interesting question, then, which is the main subject of this report, is whether feedback strategies with probing can be realized which are both simple to implement, and responsive enough to give an overall improved performance than is obtained without feedback. In particular, feedback has a tendency to introduce instabilities, and many of the issues which were the focus of this research effort revolved around the problem of controlling such instabilities.

2. THE SIMULATION MODEL

The main requirements of the simulation model are that it be simple enough to be feasibly implemented in a computer yet complex enough that end-to-end feedback strategies can be meaningfully examined. As shown in Figure 2, the modelled network consists of a central node through which pass 16 paths connecting four nodes on either side of the central node. The central node is assumed to maintain independent queues for each outgoing path. It is assumed that packet voice terminals are connected to all nodes except the central node. The traffic is specified by a matrix which indicates how many voice terminals at node i on the left of the center are in conversation with terminals at node j on the right. A fixed traffic matrix is assumed, but fluctuations in packet production due to individual talkers oscillating between talkspurt and silence are included in the simulation. Although all conversations would actually be two-way, the simulation deals only with the traffic proceeding from left to right. Thus the nodes on the left are viewed as "sender" nodes, and the nodes on the right are "receiver" nodes.

Users have available to them an embedded-coding vocoder whose characteristics are set as a parameter of the system. For most of the runs, the vocoder was assumed to be capable of synthesizing

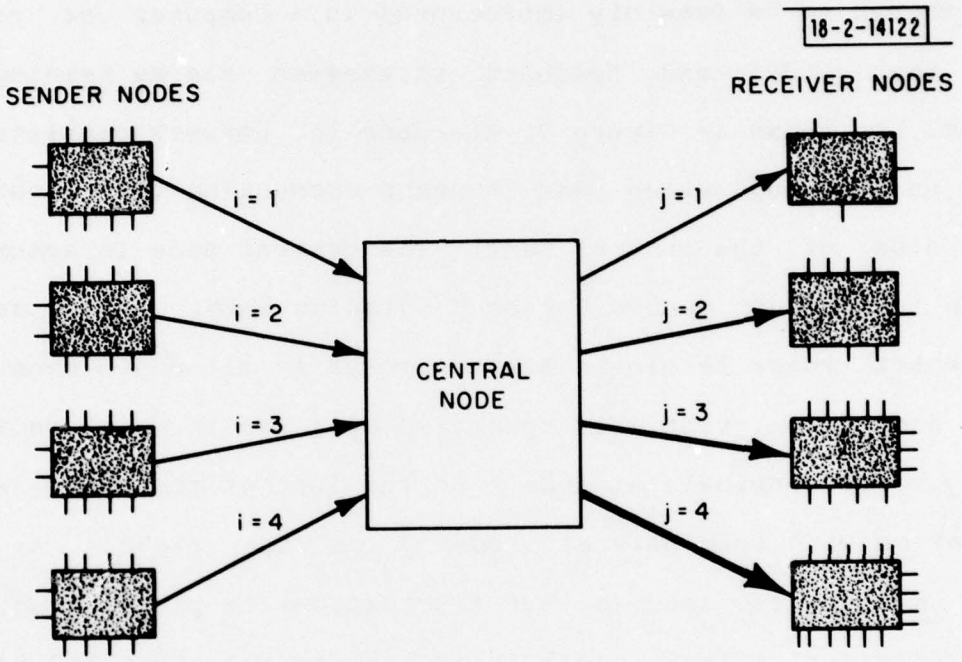


Fig. 2. Simulated network configuration. Varying thickness of receiver links symbolizes imbalanced load.

speech at 9 different rates, from 1200 bps to 10,800 bps, in increments of 1200 bps. Thus packets of 9 different priority levels are produced, and a stripping of the next higher priority level packets at a link results in a reduction of the synthesized bit rate by 1200 bps. This vocoder may be somewhat unrealistic compared with what may actually be feasible in terms of current vocoder technology. However, it is clear that if the network cannot be made to stabilize with such a vocoder, it is unlikely to do any better with one with a more restricted repertoire of possible synthesizer rates, and correspondingly fewer available priority levels.

The traffic matrix is probably the most important parameter of the system. If the load is balanced, i.e., the same fixed number of conversations are in process on each of the 16 paths, then one can expect to gain nothing from end-to-end feedback strategies. In fact, the overall performance may very well be reduced over that achieved without feedback. If the load is imbalanced, however, and especially if the imbalance is in the receiver links, then it can be expected that feedback will pay off, as will be demonstrated later.

The performance for various link strategies and feedback strategies was evaluated for a number of different traffic matrices. The matrix that was found to be most useful was one with

the same load on the four sender links but an imbalance in the load on the four receiver links, as follows:

T(ij) = number of conversations on path(ij)

$$T = \begin{matrix} 20 & 40 & 60 & 80 \\ 20 & 40 & 60 & 80 \\ 20 & 40 & 60 & 80 \\ 20 & 40 & 60 & 80 \end{matrix}$$

That is, there are 20 ongoing conversations from each sender node to the first receiver node, 40 to the second, 60 to the third, and 80 to the fourth. Thus all sender links are equally loaded, whereas there is an imbalance in the receiver links such that the first, with a total of 80 conversations, is least heavily loaded, and the fourth, with a total of 320 conversations, the most heavily loaded. The differing thicknesses in the figure of the four receiver links are meant to symbolize the imbalanced traffic load. It could be expected, with such a traffic matrix, that feedback from the 4th receiver node, in particular, to all 4 sender nodes, could cause a stripping of a significant number of bits at the source rather than at the central node, thus relieving the sender links of a useless additional load. The hoped-for result is that

users on paths leading to the first receiver node, in particular, could now synthesize speech at a higher rate than was possible without feedback. It is this traffic matrix that was used for nearly all of the runs discussed in this report.

3. IMPLEMENTATION CONSIDERATIONS

Although the conversation load in the network is fixed, the number of conversations on each path actually in talkspurt mode is a variable which is continually changing at random time intervals. One possible method to model this variable is to use Brady's [11] measurements of talkspurt and silence duration distributions from actual conversations to provide a statistical model for the talkspurt/silence alternation of each individual speaker. By aligning the time axes for all talkers on a given path, and ultimately, for all talkers in the system, it can be determined, at each instant, how many are in talkspurt mode, and hence what is the actual bit load on the system.

Another, much simpler, method to model the dynamics of the system is to ignore the identity of individual speakers and to assume a Markov approximation [12] for the sum of the activities of all talkers on a given path, even though the distribution of silence intervals and talkspurt intervals for each individual talker is not exponential. Weinstein [13] showed that this model yielded results essentially equivalent to using the Brady

statistics, as long as a sufficiently large number of talkers were modeled simultaneously. With this method, a reasonably simple model can be derived for generating random processes representative of the talker activity on each path. The process for each path is in state k(ij) whenever k(ij) of the N(ij) speakers on that path are in talkspurt mode (Figure 3a). The two rates, λ and μ, are the rate for each speaker of entering or leaving talkspurt mode. More explicitly, λ and μ are the reciprocals of the average silence duration and average talkspurt duration, respectively. Given that the system is in state k(ij), the time interval before it leaves that state is an exponentially distributed variable whose mean is the inverse of the rate of leaving that state. After the appropriate time interval, the state will change either to state k(ij) + 1 or state k(ij) - 1. The probability of an upward change is then the ratio of the upward state transition rate to the sum of upward and downward transition rates.

A further simplification can be realized by combining the speaker activity on all 16 paths(ij) into a single Poisson process (Figure 3b). This avoids the problem of having to align 16 time axes into a single time axis. A final step is then necessary to identify on which of the 16 paths each event occurred, and this can be determined straightforwardly based on likelihoods, through a procedure analogous to that used for determining the probability of an upward change. The results obtained are theoretically

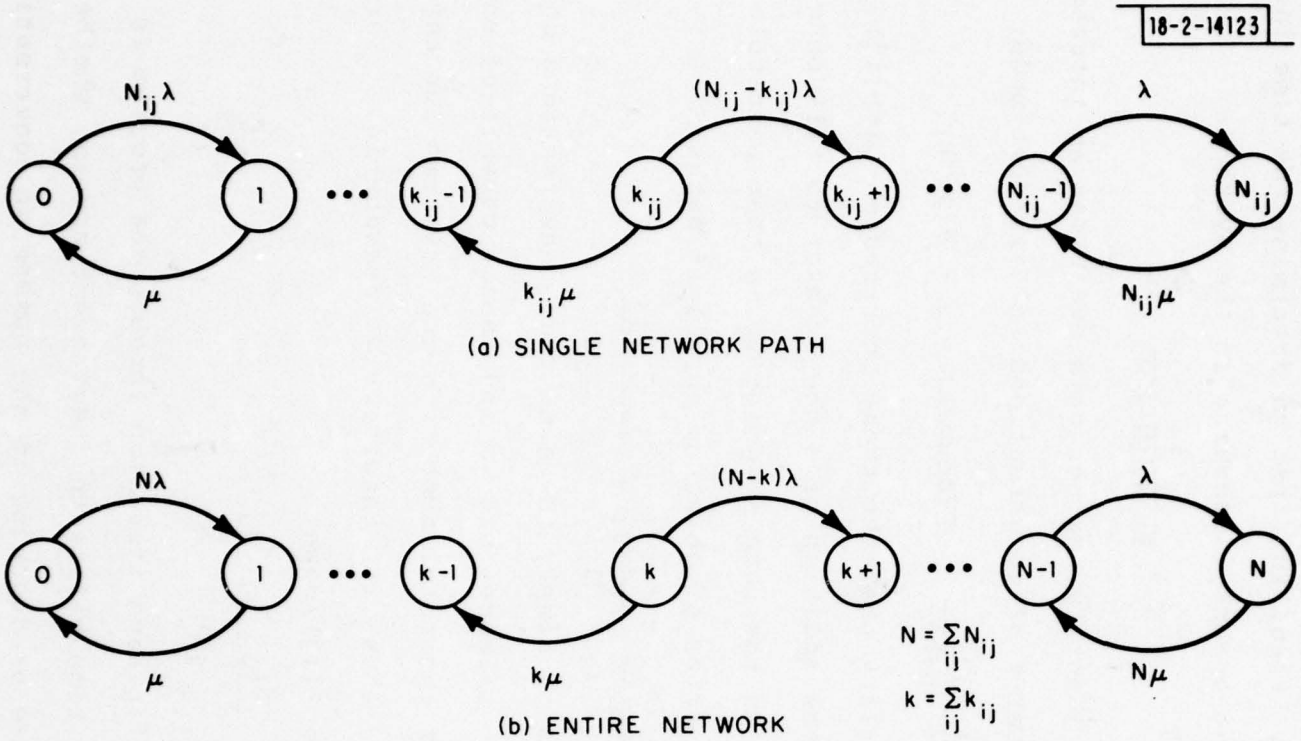


Fig. 3. (a) State-transition-rate diagram for path (ij). System is in state k_{ij} when k_{ij} of the N_{ij} speakers are in talkspurt mode

(b) State-transition-rate diagram for entire system.

$$N = \sum_{i,j} N_{ij} \qquad k = \sum_{i,j} k_{ij}.$$

equivalent to those obtained when each path is modelled separately.

The procedure, then, is first to determine the time interval before the next event occurs, somewhere in the system:

$$P(t) = \frac{1}{T} e^{-t/T} \qquad T = \frac{1}{k\mu + (N-k)\lambda}$$

The next step is to identify whether this event was an increase or a decrease of the number of conversations in talkspurt mode:

$$P(up/k) = \frac{(N-k)\lambda}{k\mu + (N-k)\lambda} \qquad P(down/k) = 1 - P(up/k)$$

Finally, the probability that the event occurred on path(ij), given that the event was the addition of a new speaker to talkspurt mode, P(ij)/up, is equal to the rate of going up on that particular path over the sum of the rates of going up on all 16 paths:

$$P(ij)/up = \frac{(N_{ij} - k_{ij})\lambda}{\sum_{i,j} (N_{ij} - K_{ij})\lambda} = \frac{N_{ij} - k_{ij}}{N - k}$$

This formula reduces to simply the number of conversations on that path currently in silence mode, and hence capable of entering talkspurt mode, over the total number in the system currently in silence mode, and thus is intuitively reasonable. The same reasoning applies to P(ij)/down:

$$P(ij)/down = \frac{k_{ij}\mu}{\sum_{i,j} k_{ij}\mu} = \frac{k_{ij}}{k}$$

The procedure for each iteration through the program is first to determine the time before the next event occurs, whether that event was an increase or decrease in the number of conversations in talkspurt mode, and on which of the 16 paths that event occurred.

Following this, the program computes the incoming bit rate (in bps) at each of the eight links. This rate depends on the number of speakers in talkspurt mode on each path that includes that link, the bit rate at which each of these speakers is transmitting bits, the current stripping threshold at that link, and, in the case of receiver links, the stripping threshold at the first link in each path.

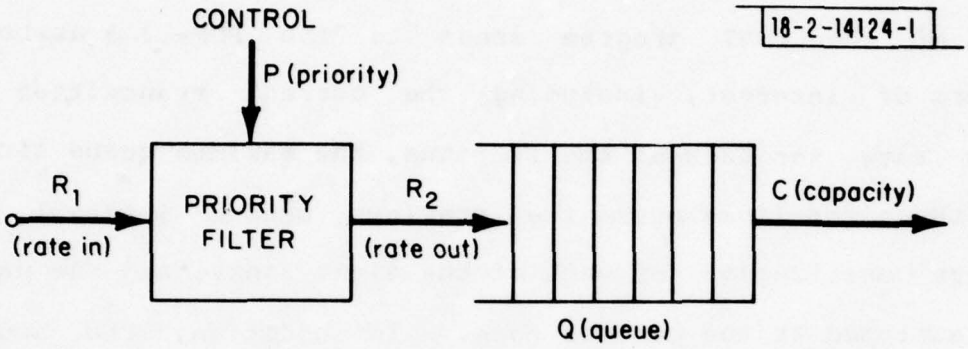
Once the incoming bit rate per second has been determined for each link, it is a straightforward process to update the queues by simply adding to the queue the product of the incoming bit rate and the time elapsed since the previous update, and subtracting the product of link capacity and time:

q = q + (rate in - capacity)*time elapsed

Negative queues are set to zero, since data cannot be sent before it arrives. At this point, a decision must be made, at each link, as to whether the current stripping threshold should be changed. Some of the strategies devised for this decision will be discussed in the next section. The final step is to update the feedback algorithm and decide whether transmitters on any of the 16 paths should change their transmission rate. The discussion of these considerations will also be deferred until a later section.

The network simulation program was first implemented in the C language on the Lincoln Laboratory PDP-11/45 facility. However, it was discovered that the program required so much computer time that only a few seconds of simulated time could be obtained even with an overnight run. Hence the decision was made to rewrite the program in Lincoln Digital Voice Terminal (LDVT) [14] code. The LDVT was capable of doing all of the calculations in better than real time, and the bottleneck in operations became the transfer of information to the PDP-11 for display and interpretation. A version of the DVT program was also created which operates in real time in conjunction with a real-time embedded-coding vocoder residing in two Lincoln Digital Signal Processers (LDSP's), with the analyzer in one and the synthesizer in the other. The vocoder was assumed to be transmitting bits along one of the 16 network paths, and the LDVT actually performed the stripping operation according to the network results, for real-time listening, as shown in Figure 4.

The important control parameters of the system are the rates λ and μ, the capacity of each of the eight links, the traffic matrix, and the rates at which the embedded-coding vocoder is capable of synthesizing speech. λ and μ were set at .77 and .83,respectively, corresponding to a mean silence duration of 1.3 seconds and mean talkspurt duration of 1.2 seconds. All of the eight links had equal capacity of .4 megabit. The traffic matrix and vocoder rates were given various settings for different simulation runs,



(1) $P = f(Q)$

(2) $P = f(R_2)$

(3) $P = f(R_2, Q)$

Fig. 4. Strategy for controlling queue at each link.

but nearly all of the discussion in this report will be with an assumed traffic matrix of (20, 40, 60, 80) for each row, as discussed previously, and with a vocoder capable of synthesizing at nine equally spaced rates from 1200 to 10,800 bits per second.

At the end of each second of simulated time, the non-real-time version of the LDVT program sends to the PDP-11 a number of parameters of interest, including the current transmitted and received rate for each of the 16 paths, the maximum queue size on each of the eight links over the previous second interval, the percentage utilization of each of the eight links, and the number of bits stripped at the central node. In addition, the program reports all changes in the transmitted and received rate for each of the 16 paths and the time at which these changes occurred. A display program is available which displays this information as a plot of rate vs time.

4. LINK MANAGEMENT STRATEGY

A network which is to make use of an embedded-coding vocoder successfully must have a strategy at each link for deciding at what priority level it should be stripping bits to control the queues. As shown in Figure 5, bits directed to a particular link come into a node at rate R1, a priority filter reduces this rate to rate R2, by discarding packets of priority less than P, and a queue builds up based on the relationship between R2 and capacity. The goal is

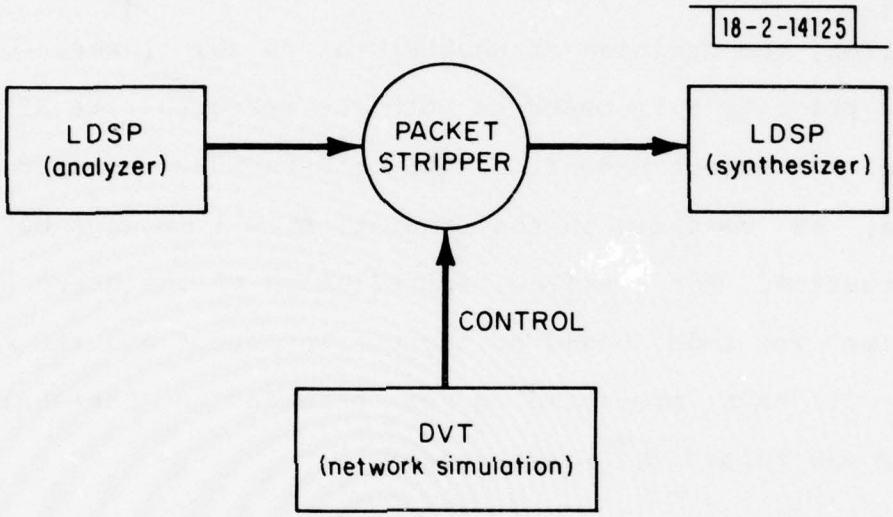


Fig. 5. Real-time system for evaluating embedded-coding vocoder quality when received rate varies over time according to network stripping strategy.

to keep the rate R2 as close to link capacity as possible, while maintaining a small enough queue size such that delays are always kept under a fraction of a second.

In the simulation, the priority P is updated at random time intervals whenever the number of users in talkspurt mode changes. For each link, the decision of whether to raise, lower, or keep fixed the priority P is based on both the measured rate R2 and the queue size. To detect underflow, only the rate R2 was used, since the queue, as computed in the simulation, will always be zero in such a situation. For overflow, a prediction of the queue at some future time was made, based on the current queue and the measured rate R2. If this predicted queue exceeded a threshold, the priority P was raised (R2 was reduced).

The simulation has available to it at any instant in time a number which is precisely the instantaneous bit rate per second. However, this number would not in general be available to a node in a real network, and hence, to be more realistic, it was decided to have each node measure the rate R2 over a fixed time interval. The measuring process introduces a time delay before the node is aware of a significant change in rate, and this delay necessitates a reduction in efficiency of about 5%, compared with that realized using the exact R2, in order to avoid catastrophic dropouts.

The strategy used for controlling queue sizes was as follows: R2 was measured over a 100 msec period. If at the end of that period, R2 was found to be less than a certain percentage, Θ_L, of the capacity of the link, the stripping priority P was lowered (more bits were allowed through). If, on the other hand, the projected queue after another 100 msec period, based on the current queue and the measured R2, was greater than another percentage, Θ_U, of the capacity of the link, the stripping level P was raised (fewer bits were allowed through). In addition, as an emergency measure, if the queue itself became greater than 10% of the link capacity at any time, the priority P was raised.

It was found, through several experimental runs, that the performance of the stripping strategy was critically dependent upon the two numbers, Θ_L and Θ_U, the thresholds at which P was lowered or raised. For example, Θ_L could be set low enough (say 60% of capacity) such that the system would finally settle at a steady state condition where the rate fluctuated between 65% and 100% of capacity, never crossing either threshold. In this case, the stripping priority P would never change, and therefore, the received rate would remain fixed.

However, with such a low threshold the link is likely to be underutilized. If the threshold is set higher, say at 80% of capacity, then there is a much greater likelihood of R2 crossing

the threshold due to statistical fluctuations in the load. If, for example, the mean rate is 95% of capacity, the measured rate would be much more likely, statistically, to reach 80% than 60% of capacity. The result would be an increase in the number of bits accepted through the priority filter, which would quite likely drive the projected queue over the threshold Θ_U, forcing the rate back down again. On the other hand, the higher threshold should result in an overall higher utilization on the link.

We experimented with several settings for the threshold Θ_L, and found 70% to be the best choice for maintaining a more or less steady-state performance without an excessive sacrifice in utilization. Figure 6 shows the results (received rate vs time) for the path i=1, j=1, for the network with the imbalanced load (20, 40, 60, 80) with two settings for the threshold, .7 and .75. It can be seen that with a setting of .7 the received rate stays steady as a function of time, whereas with .75 there are occasional dropouts due to overloads.

The issues for the threshold on the projected queue are similar. If the threshold is set near zero, it is likely to be crossed too frequently. If it is set too high, however, intolerable delays are likely to build up. The number 10% of capacity represented a compromise which kept delays under 100 msec, but allowed queues to build up somewhat over short term intervals without causing unnecessary changes in stripping level.

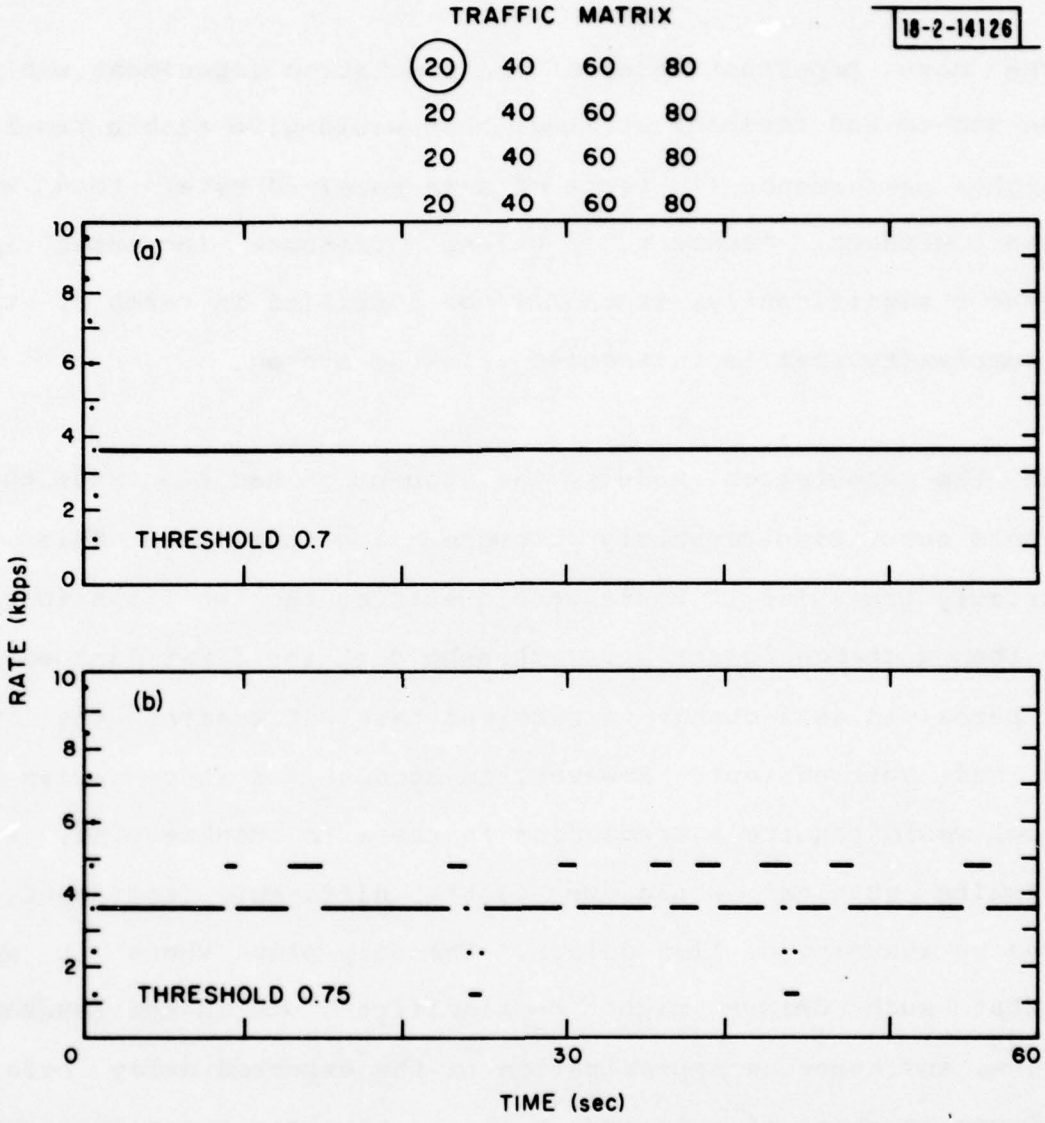


Fig. 6. Comparison of performance with no feedback for two different settings for θ_L. Plot is of received rate vs time for the path i = 1, j = 1.

5. FEEDBACK STRATEGIES

The most important goal of the simulation experiment was to find an end-to-end feedback strategy that would give stable results with higher performance (in terms of mean received rate) than was obtained without feedback. Unless feedback increases the performance significantly, it cannot be justified in terms of the added complexity that is introduced into the system.

In the simulation models, the assumption had been made that all events occur simultaneously throughout the network. This is not strictly true, for if there were queues at the two links in the path, then a change in stripping threshold at the first link would not be perceived as a change in received rate until after the two queues had spilled out. However, to account for these delays in the model would require a tremendous increase in bookkeeping, and the results obtained would be little different from what is obtained by assuming no time delays. The only place where it was felt that such delays might be significant was in the feedback algorithm, and hence an approximation to the expected delay before an analyzer is aware of a change in the synthesizer's received rate was included in the simulation. This delay was modelled as twice the time it would take to spill out both queues in the path, and is meant to represent a round-trip path, where the phantom return path

is assumed to introduce the same time delays as the forward path.

Two feedback strategies, "continuous probe" and "periodic probe", that were tried initially are obvious methods that appear reasonable at first but are not effective in practice. With the continuous probe strategy, users always transmit at the next higher rate than the reported received rate, and hence can immediately respond to a decrease in the network load. However, this quick response is achieved at the cost of always having to transmit more bits than are actually being received. Unless the traffic matrix is extremely imbalanced, the method yields little or no improvement in received rate over that realized without feedback.

The periodic probe method was an attempt to realize greater gains, but it exhibited undesirable instability problems. For this strategy, users transmit at the reported received rate. However, if, over a fixed time interval (usually on the order of one second) no bits have been discarded in the network, the transmitter attempts a probe. He begins transmitting at the next higher rate, and does so until he obtains the receiver's report on his received rate. If the probe was successful, he continues transmitting at the higher rate. If not, he drops back to the low rate.

The results for this feedback strategy are shown in Figure 7, for a path which included the lightly loaded receiver link, given

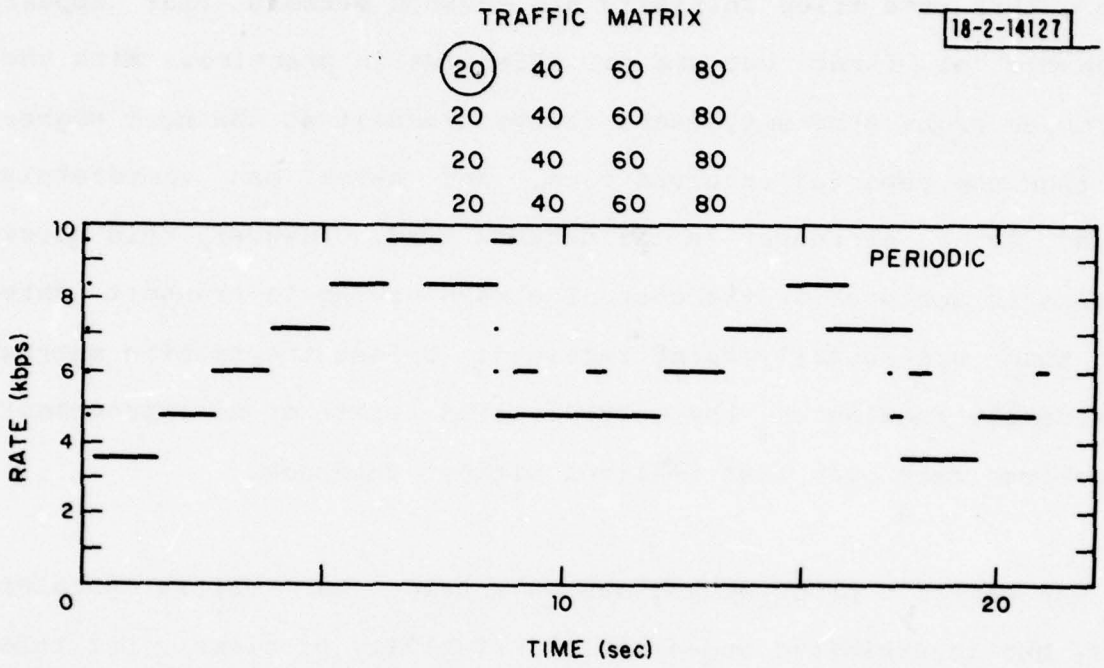


Fig. 7. Transmitted and received rate vs time for periodic probe feedback strategy, for path i = 1, j = 1.

an imbalanced traffic matrix (20, 40, 60, 80, discussed previously). The path is from the first sender node to the first receiver node (i=1,j=1). It would be expected that feedback from the fourth receiver node would cause the corresponding transmitters to reduce their transmission rate, which would allow the first sender link to accept more bits through for users directed to the first receiver link. Since the first receiver link is lightly loaded (20x4 conversations), it is to be expected that the bottleneck in the path would be the sender link, and hence by a lightening of the load on this link through feedback an overall higher received rate would be possible at the first receiver link.

In the figure, both transmitted and received rate are shown as a function of time. Wherever there are two values for rate, the higher one is the transmitted rate. Transmitted rates and received rates for this method are usually equal, except in the case of an unsuccessful probe. This means that almost no bits are being discarded at the central node. The mean rate is significantly higher than that obtained with no feedback. However, there seems to be a pattern of a staircase-like rise followed by a rather sudden collapse in the received rate. Such rapid fluctuations in rate are not likely to be desirable for producing high quality synthesized speech; nor would they be expected to produce the highest possible mean received rate.

This pattern arose as a consequence of several factors, which may be understood by frequent reference to Figure 8. In Figure 8 is a schematized graph of received rate vs time for two paths sharing the first sender link. The staircase rate is the pattern observed for the path to the lightly loaded receiver link; the other pattern is the rate observed for the path to the heavily loaded receiver link. In the latter case, probes are always unsuccessful, due to the heavy load on the receiver link. Probe bits do get through the sender link, however, and since a large number of users are probing, they represent a heavy additional load on that link for the duration of the probe. At t = Ø, the simulation initializes with all links accepting all priority bits. There is a rapid initial collapse in the rate, as queues build up, and users respond by immediately lowering their transmitted rate to the low crisis rate on the link. Once the queues spill out, the sender link is underutilized, and hence probes are generally successful. The result is that on the average the load on the link steadily increases. At the beginning, when the link was only being utilized to 60% of capacity, the short probe by users of the 4th receiver link did not pose a problem to the first sender link. However, when the load on the sender link was very near capacity, the same probe by the numerous users on the heavily loaded receiver path introduced a large excess load on the sender link which pushed its input rate to far exceeding capacity. The link had to discard bits, but it could not discard the probe bits because they were of

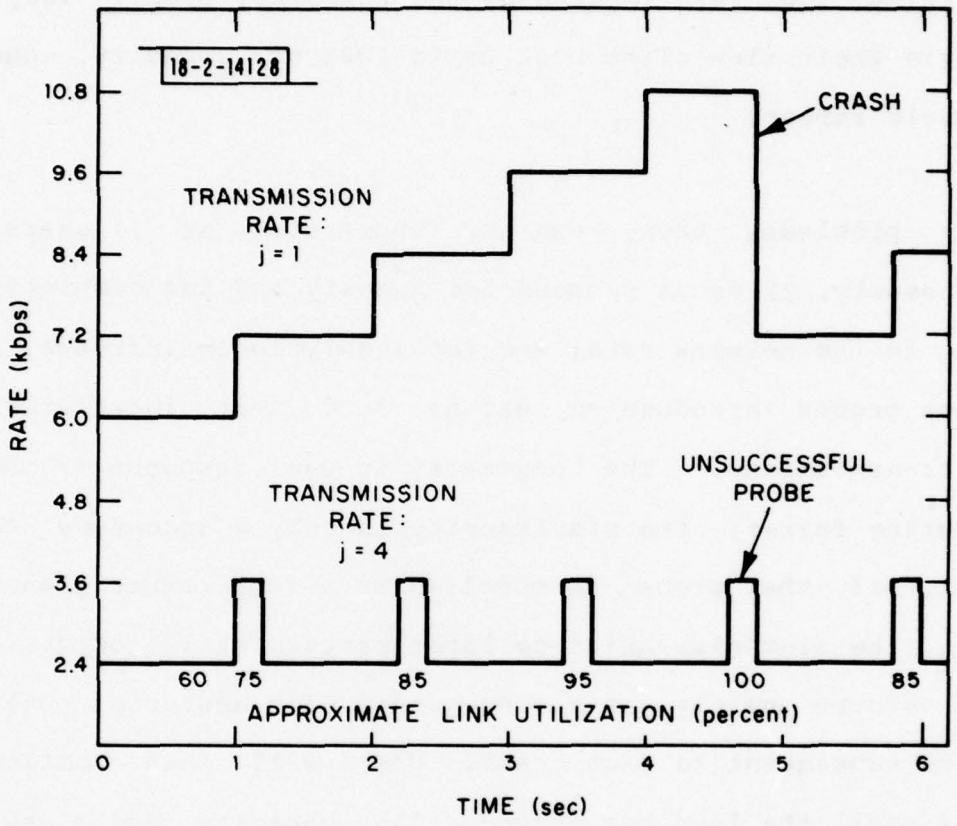


Fig. 8. Schematized plot of transmission rate vs time for two paths sharing first sender link and of percent utilization on first sender link to illustrate instability phenomenon with periodic probe strategy.

high priority. Hence it had no choice but to collapse the rate of those few users directed to the lightly loaded receiver link. These users responded by setting their transmission rate to the artificially low rate imposed by the temporary probe. They would then begin their slow climb back up to 100% of capacity, and the cycle would repeat.

The problems, then, can be enumerated as 1) users probe simultaneously, 2) users respond too quickly and too completely to a drop in the network rate, and too slowly to an increase, and 3) fruitless probes introduce an excess load that interferes with valid transmissions. The asymmetry in user response is the main contributing factor; the simultaneity is only a secondary factor. In fact, if the probe is modelled as a ramp rather than a step function, the slow rise/quick collapse pattern still occurs. The problem occurs mainly as a consequence of transfer of control to the users subsequent to each crash. Users will then continue to up-probe until the load has exceeded link capacity, and a new crash will ensue.

To solve the problem, an entirely new strategy was developed, for which user response is sluggish and symmetrical, users no longer act in unison, and fruitless probes are eliminated. To eliminate fruitless probes, a new probing strategy was developed in which the network actually writes into a field in all priority one

packets information concerning the highest rate that could have gotten through. The result is effectively as though the transmitter had transmitted packets of all priority levels, and the synthesizer had reported back the lowest level that was successfully transmitted. However, instead of having to send nine packets to obtain this information, the sender would only have to transmit a 4 bit field with the number 9 written into it (Figure 9). Each link in the path would replace the number in the field with the stripping priority, P, if P was less than the number currently in the field. Such a test-and-replace operation requires very little processing overhead at each node, and the amount of excess data that must be sent for probing is thus reduced to only a 4 bit attachment to priority one packets.

In the figure, two such fields are indicated, and the network writes into only the left-most field. When the receiver receives this information, he has no use for it, but instead simply transfers it to the right-most field of the priority one packets that he is transmitting in the return path. Eventually, then, the loop will be closed, and the transmitter will know what rate the network can support. A similar operation is occurring, meanwhile, in the reverse path, and P2 in the figure is the lowest priority that the reverse path can currently accept.

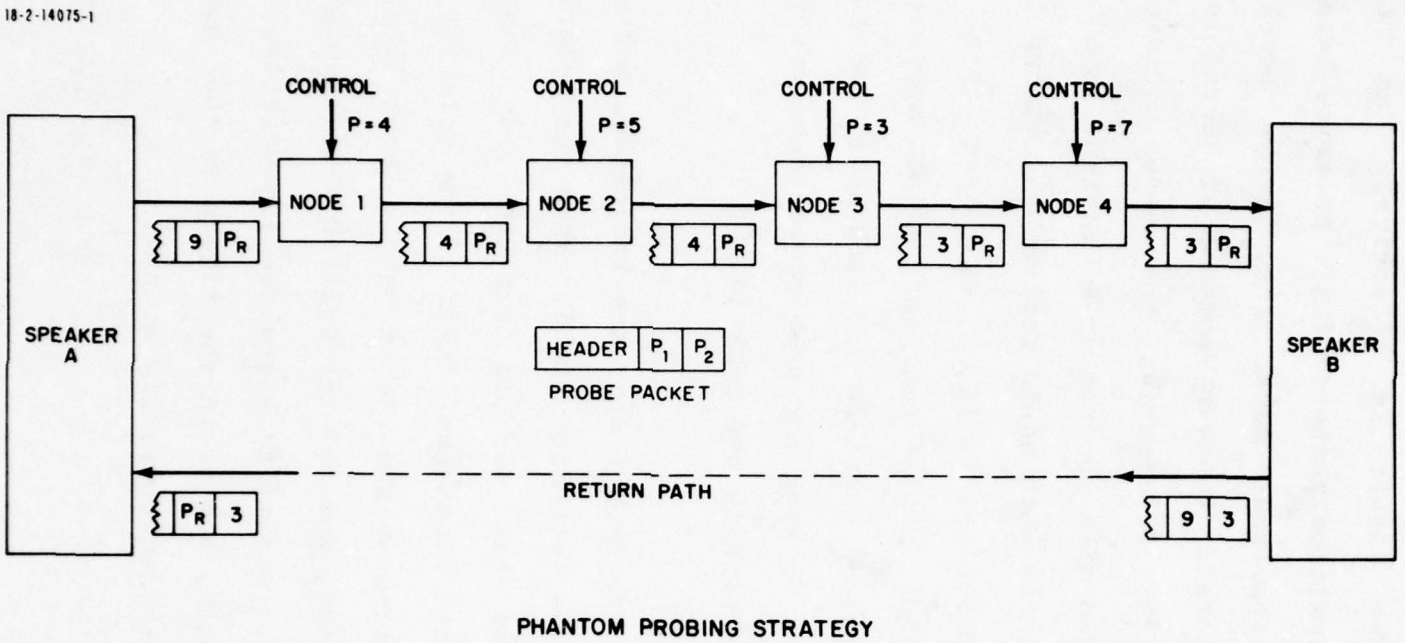


Fig. 9. Phantom probing strategy to eliminate fruitless probes.

An additional change in the new feedback strategy as compared with the old one is that users are much more sluggish in their response to network changes, and do not exhibit an asymmetry in their response to a reduction in the rate the network can accept as opposed to an increase. In the new strategy, each user is allowed to change his rate at periodic time intervals, where the period, equal for all users, is on the order of 5 seconds. If, at the end of his period, a transmitter finds that the network can accept a higher rate than that at which he is currently transmitting, he responds by increasing his rate by one level. Likewise, if his rate is higher than that which the network can currently support, he decreases his rate by one level. Otherwise, he makes no change, until the time of his next update.

A final modification is that the users no longer act in synchrony. One could imagine, in a real network, a strategy whereby each user's first period begins at the time of dial up. Since users will dial at random time intervals, this will result in a randomization of the times of rate change for the various users on a given path. Since it is not feasible in the simulation to treat each conversation as a separate entity, a simplification was made such that it is assumed that a certain percentage of the users have probed after the same percentage of the probing interval has been exhausted.

The results for a simulation run using the new strategy are shown in Figure 10, for the same path (i=1,j=1) with the same traffic matrix (20, 40, 60, 80) as was used for the previous examples. Where formerly only 20 seconds of simulated time were shown, this time the run was carried out for 120 seconds. Only the received rate is shown, and the fractional rates are mean rate per customer rather than realizable rates. It can be seen that the system is much more sluggish, with a steady state condition being reached only after 35 seconds. However, the received rate remains steady for the remainder of the run, a much more pleasing result than was obtained formerly. Furthermore, the mean received rate is significantly higher than that obtained with no feedback. Overall performance is considerably better (in terms of number of bits stripped at the central node and mean percent utilization on the links) than was obtained using either of the other two feedback methods.

The overshoot exhibited at around 30 seconds is characteristic, and occurs as a consequence of the fact that user actions are more sluggish than link actions. Whenever the link can transmit all of the bits R1 and still occasionally cross the threshold Θ_L, the link will eventually lower the priority P to the lowest level. Users will then continually discover that they can raise the transmission rate, and will do so until finally the link must reduce P as a consequence of the projected queue exceeding Θ_U.

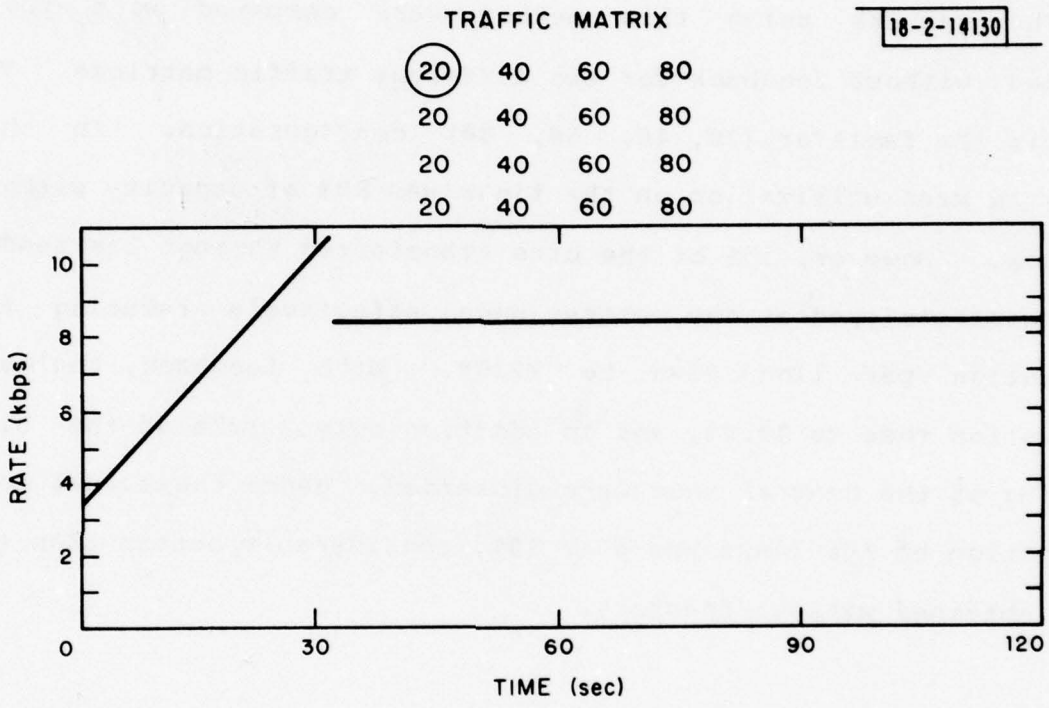


Fig. 10. Received rate vs time for the path i = 1, j = 1, using sluggish feedback strategy. Rate is mean rate per customer rather than realizable vocoder rates.

The link effectively transfers control to the users, and thus the increase in R2 is no longer controlled by Θ_L, and will proceed until finally the overload condition is reached.

The results using this method were compared with those obtained without feedback for two different traffic matrices. The first is the familiar (20, 40, 60, 80) configuration. In this case, the mean utilization on the links was 80% of capacity without feedback. However, 15% of the bits transferred through the sender links were stripped at the central node, effectively reducing the utilization per link down to 72.5%. With feedback, the mean utilization rose to 90.4%, and in addition only 2 1/2% of the bits arriving at the central node were discarded. Hence the actual mean utilization of the links was over 89%, considerably better than the 72.5% obtained without feedback.

The other traffic matrix tried was the transpose of the above matrix: thus the receiver links were all equally loaded, but the sender links were imbalanced, with the first one being the most lightly loaded. In this case it would be anticipated that feedback schemes could offer no advantage over no feedback, since bits would be stripped at the input of heavily loaded sender links, rather than at the central node. In fact, feedback would tend to reduce the overall performance since senders could not immediately correct for an improvement in the rate the network could accept.

The overall utilization was found to be 91.4% with no feedback, and only 88.9% with feedback. However, the surprise was that the received rate as a function of time for paths including the lightly loaded sender links was very steady if feedback was used, but tended to incur dropouts without feedback (Figure 11). The reason for this phenomenon is that the feedback scheme operates as a smoothing filter to control fluctuations in the rate R1 arriving at each receiver link as a consequence of actions in the sender links. For instance, when the priority level P is decreased by one level at a particular sender link, the resulting increase in the load is felt immediately at the receiver links if there is no feedback, whereas a more gradual increase in the rate would be observed if feedback is being utilized, as users one by one become aware of the increased network rate. The result is that, without feedback, receiver links are likely to have to cut back the accepted rate for users of the lightly loaded sender link in order to handle an overload introduced by a sudden increase in the rate getting through a heavily loaded sender link. The problems are somewhat analogous to those introduced by the "periodic probe" feedback strategy, except that recovery from sudden drops in the rate is much faster without the blanking period restriction.

6. INCLUSION OF DATA TRAFFIC

It would appear intuitively that an embedded-coding vocoder could operate quite successfully in a network environment where the

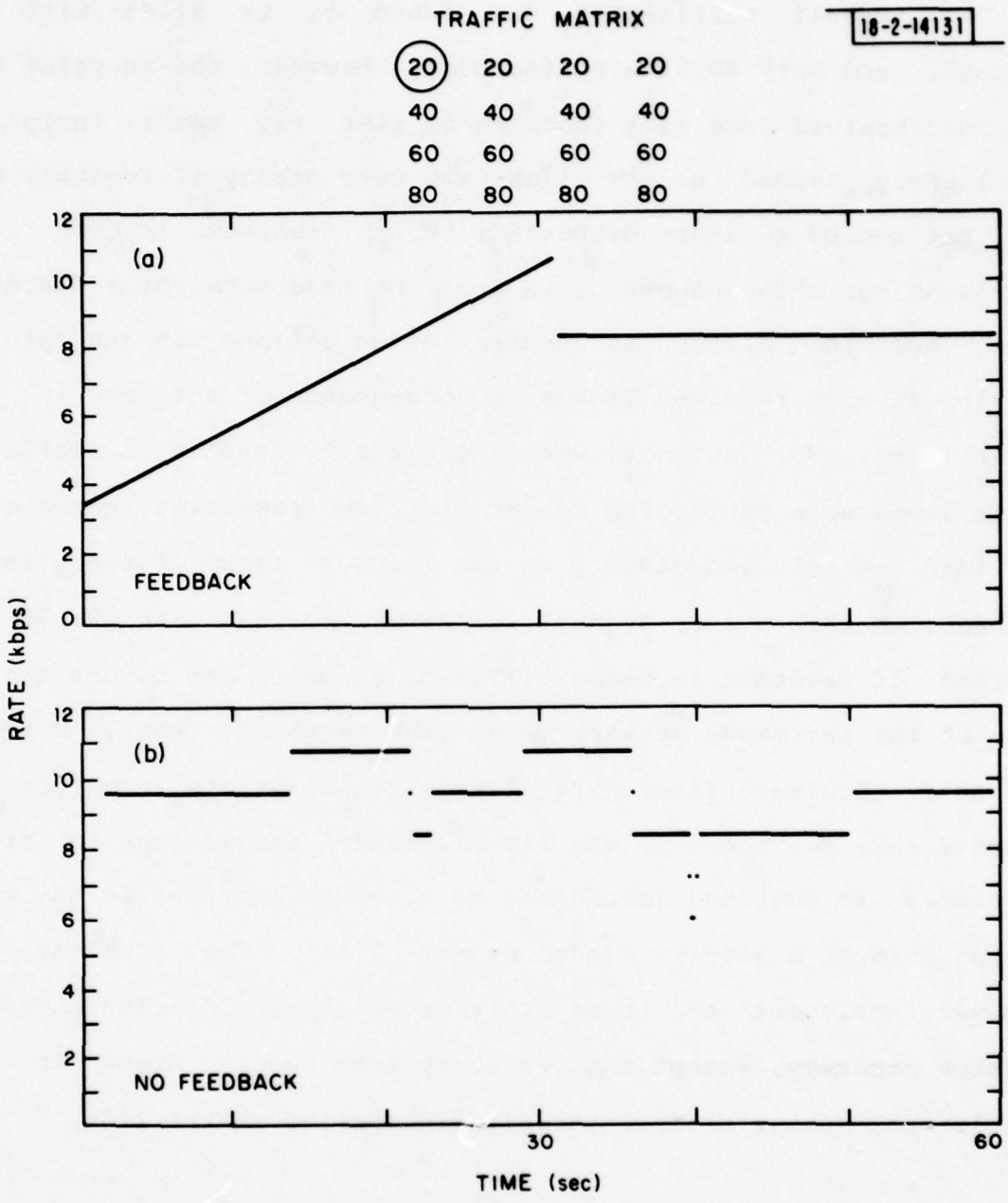


Fig. 11. Received rate vs time for path i = 1, j = 1, for transposed traffic matrix (imbalance is in sender links)
(a) with feedback
(b) without feedback.

link was being shared with data. The expectation would be that an increase in the data load could be dealt with by dropping the vocoder rate, and a potential sudden dropout in vocoder rate due to a temporary overload could be averted by temporarily allowing data queues to build up.

The first issue that must be resolved in combining data and voice is a strategy for portioning out the link to the two types of users, who have different requirements in terms of delays and stripping. In particular, speech users cannot afford long delays, but are only minimally disturbed (through degraded vocoder quality) by losses of all but priority one packets. With data, on the other hand, the only restriction on queue buildups is available buffer space in the node, but none of the bits should be discarded.

A possible solution would be to give all data packets a priority of one, which will assure that they are not discarded. However, the network will have no way of knowing that data packets can be delayed, and hence may insist on rushing these packets through at the expense of lower priority speech packets.

Another solution is to keep the data in a separate queue, for which there are less strict requirements on maximum size. Fluctuations in the data load could then be smoothed out at the point of exit on the link by allowing the queue to expand and shrink as needed.

For the experiment, the data was modelled as a stream of packets arriving at exponentially distributed time intervals. All data packets were of fixed size, 1200 bits, the same size as the speech packets. The capacity of all eight links in the system was doubled (from .4 to .8 Megabits/sec), and the mean data load was set to equal exactly the additional capacity on each link. Thus, the speech users now have available to them a network with the same mean capacity as formerly existed, but with a capacity which effectively varies with time, as the data load fluctuates.

The strategy for apportioning the link to speech and data need depend only on the data traffic, since speech bit rate can be adjusted by stripping off lower priority packets as needed. Although for the simulation the mean data rate is a fixed known number, in general each node would have to predict the data load based on some past statistics. In the simulation, the mean data rate is measured for a fixed time interval and the value obtained is used in conjunction with the data queue as an estimate for deciding how often to send data packets out on the link for the next fixed interval. Hence the model is designed to be able to deal with a data load whose mean is an unknown variable rather than a known constant, and thus represents a more realistic system.

Once it has been decided what percentage of the packets should be data for a given time interval, a strategy is needed for

actually realizing that percentage without ever holding up either queue for a long time. One possibility (Figure 12) is to consider a small number, M, of packets as a unit, and to send data and speech alternately until the one with the lower allocation has met its quota. The other type is then sent exclusively for the remainder of the set of M. In addition, if the speech queue is empty but instructions are to send speech, the node would send data rather than let the link go idle. The same would hold for the reverse. Thus only if both queues are empty will the link remain idle.

The strategy which was developed was to measure the incoming data rate over a one second interval, and to set the percentage of the link devoted to data such that, if the same number of bits were to arrive during the next second interval, the data queue would dwindle to exactly zero by the end of the next second. This fraction was then truncated to 4 bits, or 16 levels, thus assuming M = 16. Truncation was thought to be preferred over rounding since it was not necessary for the data queue to actually reach zero by the end of the next second. Thus with truncation the data will be allocated a capacity somewhat less than the estimate, where the estimate is generally somewhat more than is actually needed. When the allocation turns out to be inadequate, the queue will have built up by the end of the next second, at which time a larger portion of the link will automatically be given to data, to compensate.

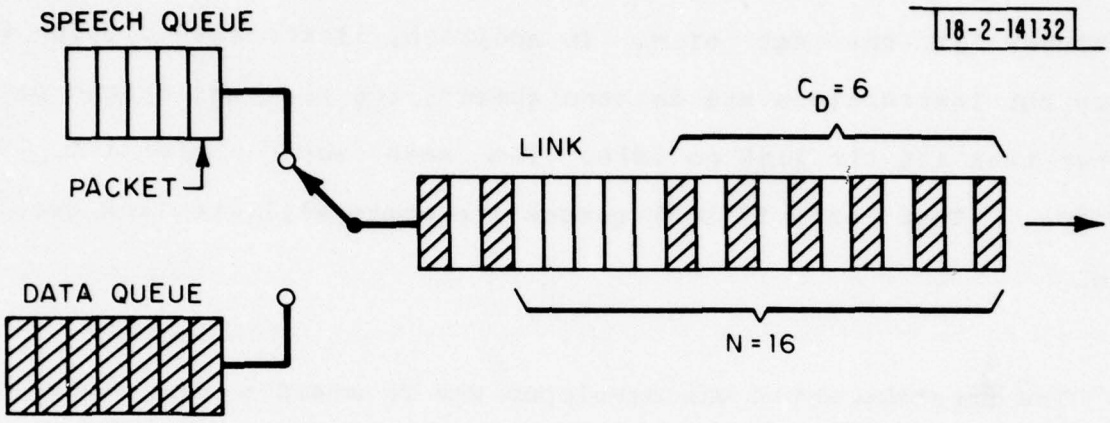


Fig. 12. Strategy for portioning out link to speech and data.

The strategy for speech, meanwhile, is the same as before, with the packet stripping priority being based on the ratio of the incoming rate R2 and the link capacity, C, as well as the queue size. The only difference is that C, the allocated capacity after data has been accounted for, will now change as a function of time, and therefore the predicted queue will be less accurate than formerly. The true capacity will not only be readjusted every second, but will also fluctuate within the one second intervals, because of the fact that the speech queue can spill out whenever the data queue is zero. The consequence is that there is a greater fluctuation in the ratio R2/C, and the computed ratio will not in general equal the true ratio.

The results are shown in Figure 13 for the path i=1, j=1, using the imbalanced traffic matrix. Speech users are assumed to make use of end-to-end feedback. With the original settings for Θ_L and Θ_U (70% and 10%) (Figure 13a), there are frequent large dropouts in the received rate, as a consequence of overload on the links. Not only is the threshold setting too high because of the greater fluctuations in the ratio R2/C, but also the crashes are more severe because of the interplay of speech and data. As long as the speech load is below, on the average, the capacity reserved for speech, the data can obtain more capacity than has actually been reserved for it because data gets the speech spillover. When the speech bits begin to consume their entire allocation, however,

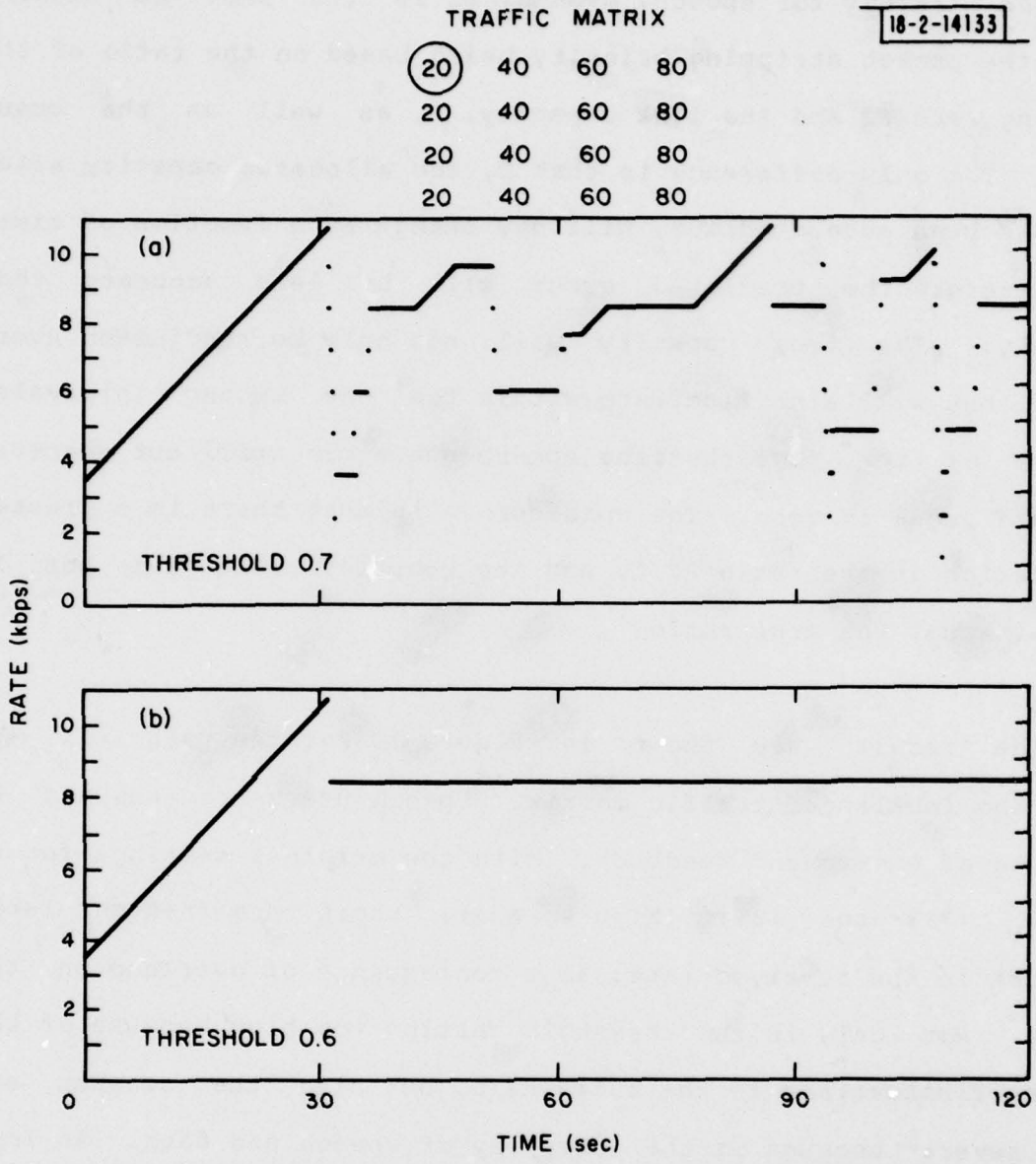


Fig. 13. Received rate vs time for system with 50% data load; data allocation is truncated; i = 1, j = 1
(a) θ_L = 70%
(b) θ_L = 60%.

the data no longer gets extra capacity, and hence its queue begins to build up. A buildup in the data queue will result in an increased allocation for data, which will shrink the portion reserved for speech at a time when such a cutback can be ill-afforded. The result is a sudden drastic drop in the received rate.

This problem is a more extreme example of the basic problem that has haunted the system from the beginning. The problem is that when input rate exceeds capacity for even a short time, queues build up very quickly to intolerable proportions. The obvious solution in such a situation is to cut way back on the rate temporarily, to let the oversized queues spill out.

A better alternative is to avoid such a situation by not allowing the stripping priority P to decrease if such a decrease may result in overload. This means a reduction in the threshold Θ_L. Indeed, if Θ_L is reduced to 60%, the performance is again comparable to that obtained without data, i.e., dropouts are very infrequent, as shown in Figure 13b. However, even for this run there are two instances, one in paths through sender link 2 and the other in paths through sender link 4, when the rate dropped suddenly to only 2400 bits per second. Hence the frequency of the sudden dropouts has been greatly improved, but their severity, when they occur, remains the same.

A change in the threshold Θ_U will not help the situation, for queues will still build up rapidly, and the same sort of crash will occur when the projected queue crosses a lower threshold as occurred when it crossed a higher threshold. In fact, a lower threshold would be crossed more frequently due to statistical variations than would a higher threshold, thus aggravating the problem. The preferred setting for Θ_U was found in all cases to be the maximum affordable level to keep speech queues within the 10% tolerance range (keeping delays under 100 msec).

An alternative strategy to alleviate catastrophic dropouts is to give the data a larger portion of the link capacity than it needs such that data queues will not build up excessively. This could be arranged by rounding rather than truncating the number computed for the fraction of the link to allocate to data. Since data is always allocated a portion greater than it actually needs, data queues will not build up, even in the event of overload of speech. Then a strategy could be devised for temporarily reallocating the data portion (say cutting its allocation by 50%) whenever a crash is imminent. The data queue could then be viewed as a reserve resource for getting over a heavy speech load condition, to avoid having to suddenly drop the speech rate to a very low level.

The results for such a strategy are shown in Figure 14. Here the thresholds Θ_L and Θ_U are kept at 70% and 10%, as in Figure 13a and in the case of no data. However, the data is given a rounded rather than truncated portion of capacity. Furthermore, whenever the priority threshold P needs to be increased, the data portion is simultaneously cut to 50% of its former value, for a 100 msec period. Such a cut gives the speech a chance to spill out its temporarily overloaded queue, at the expense of an affordable buildup in the data queue.

As can be seen, there are several tradeoffs involved here, with a slightly higher efficiency being realized at the expense of more frequent dropouts in received rate. Although the rounding strategy for data results in a somewhat lower performance in terms of efficiency than was obtained with truncation and a 60% threshold (91.5% vs 92.2%), there is a much smaller danger of drastic dropouts occurring, even when the threshold is set too high, i.e., the setting of the threshold is less critical when a conservative allocation is used for the data. Furthermore, the desired threshold setting is more closely matched to the optimal setting when no data is present. Certainly it is preferred not to require a threshold setting that is a function of the mean data load.

Figure 15 shows the data queues and efficiency as a function of time for the eight links for the first 30 seconds of the run

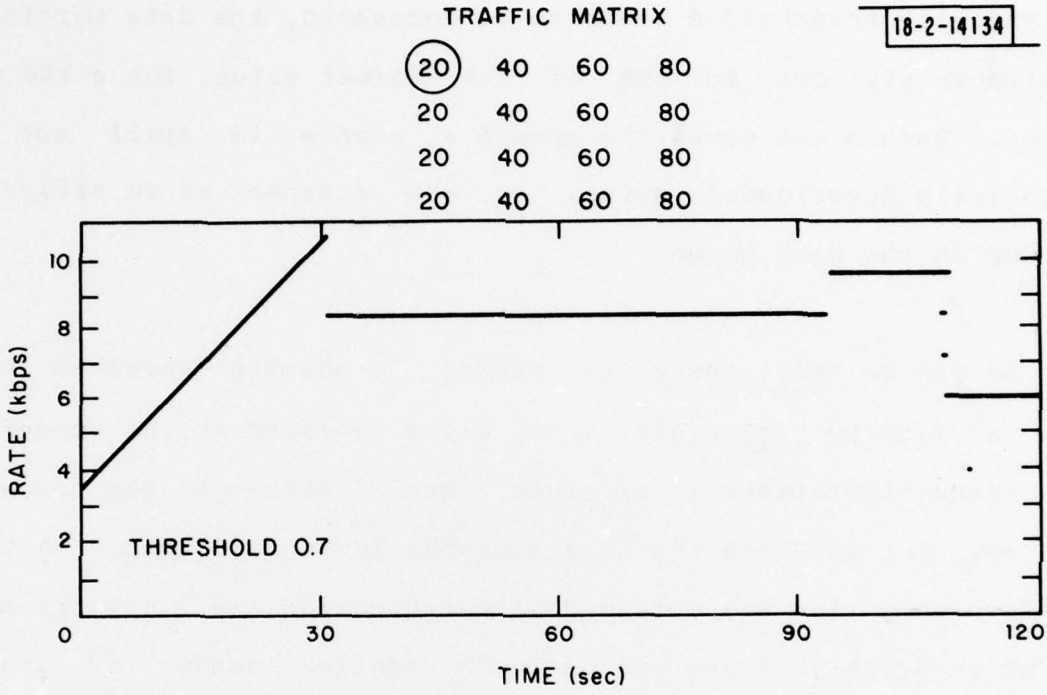


Fig. 14. Performance with 50% data when data allocation is rounded, and reduced by 50% for 100 msec whenever speech rate drops. θ_L = 70%.

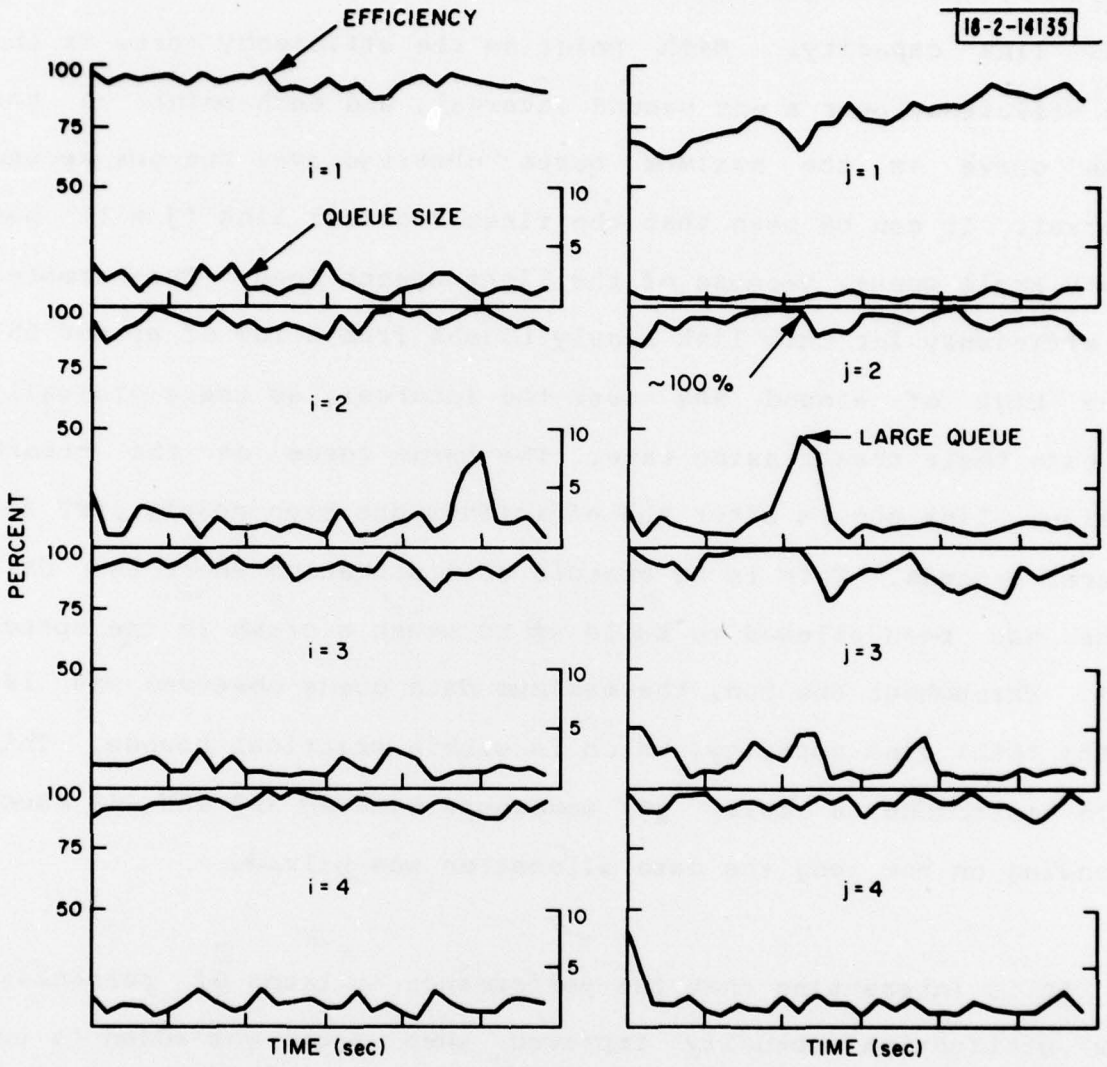


Fig. 15. Efficiency and data queues at all eight links for first 30 seconds of run corresponding to Fig. 14a. Scales are different for two functions. Data queue expressed as percentage of total link capacity.

corresponding to Figure 14. The scales for the two functions are different, and data queue size is expressed as a percentage of total link capacity. Each point on the efficiency curve is the mean efficiency over a one second interval, and each point on the queue curve is the maximum queue observed over the one second interval. It can be seen that the first receiver link (j = 1) has a very small queue, because of the light speech load. Furthermore, the efficiency for this link slowly climbs from a low of around 65% to a high of around 90% over the interval, as users gradually increase their transmission rate. The large queue on the second receiver link occurs after the efficiency has been nearly 100% for several seconds. This is an example of a situation where the data queue has been allowed to build up to avert a crash in the speech rate. Throughout the run, the maximum data queue observed was 10% of the total link capacity, which is within practical bounds. This would introduce a delay of somewhere between 200 and 400 msec, depending on how long the data allocation was halved.

It is interesting that the performance in terms of percentage link utilization actually improved when data was added to the system as compared with the performance with speech only (91.5% vs 89.1%). Such improvement is probably a consequence of the fact that the speech bits can take advantage of excess capacity in the data reserve and vice versa. However, this effect would be offset by the fact that a greater variability in the ratio R2/C is

introduced, and hence it was not immediately obvious that performance would improve.

The simulation has always assumed a fixed mean data rate and a fixed mean speech rate. In a real network users will continually enter and leave the system, and therefore the mean will change as a function of time. The initialization and termination of conversations is probably not an important effect, since the observed effect will be a slow drift in the mean number of users, which can be handled by the system. The data fluctuations are of more concern, since a user could theoretically suddenly decide to transmit an enormous file across a link. Here a strategy would be needed to control the number of data bits that could enter the system over a fixed time interval, by holding up the data at the source. The details of the solution are not immediately obvious, but will not be dealt with in this report. In any case, another reason for desiring a generous rather than sparing data apportionment is that the system is better able to cope with increases in the data load.

7. SUMMARY AND CONCLUSIONS

A network simulation program has been implemented on the LDVT-PDP-11/45 facility which has addressed several issues with regard to the feasibility of using an embedded-coding vocoder in a network environment. Although many assumptions were made for the

simulation that are not strictly valid in a real network, it is believed that the results obtained will be of use when an attempt is made to incorporate an embedded-coding vocoder into a voice/data network.

The major problem that occurred, and which reappeared at every stage of the simulation, was the tendency for the received rate to suddenly drop by several priority levels at times of overload. This problem occurred even without feedback, if the setting for Θ_L was too high. Whenever statistical fluctuations in the measured rate are such that the threshold Θ_L will occasionally be crossed even when the mean rate R2 approaches capacity, then such dropouts will be expected to occur.

With a feedback strategy which is immediately responsive to drops in the network rate, but sluggish in its return to a high rate, such as was the periodic probe algorithm, the observed received rate will be a repetition of a staircase-like rise followed by a sudden collapse. Each such crash is unavoidable because the network has passed control over to the users subsequent to the previous crash, and the users will probe until the upper limit Θ_U is exceeded, without the benefit of the restrictions ordinarily imposed by Θ_L.

The preferred feedback strategy is one where user response is very sluggish, and symmetrical with respect to rate increases and rate decreases. Each user is only allowed to change his rate once every few seconds, and even then by at most one priority level. Because the strategy is so sluggish, users are not capable of crowding the threshold Θ_U with a rapid increase in rate. Nor will they suddenly set their rate to a low level subsequent to a network rate collapse, should such a collapse occur. The only possible disadvantage is that users take a long time to adjust to a large sustained change in the network rate. The hope is that network load will change slowly enough such that users can more or less track such changes. If not, the result is not catastrophic, but only a reduced efficiency (if the network can suddenly support a higher rate) or a greater stripping of bits at intermediate nodes (if the network must suddenly drop its rate).

The incorporation of data initially introduced more serious collapses in the speech rate when the threshold Θ_U was crossed, but the problem was greatly alleviated by the modified strategy which kept the data queues low except in times of imminent crashes. It was concluded that a preferred algorithm is to give the data more capacity than it actually needs except during times of trouble in the speech domain, both to alleviate crashes and to allow some flexibility if data loads should increase.

The scope of this simulation program was somewhat restricted, and there are many aspects of the problem which could be further explored through a more complicated simulation experiment. The simulation made use of only a two-hop network, did not allow for any changes in the mean rate either for data or for speech, investigated only a small set of different traffic matrices, assumed an idealized vocoder that is probably better than what will actually be available, and made many assumptions (such as Poisson distribution) that are not strictly valid. Even with these restrictions, however, it was found that effects were often unanticipated and difficult to interpret. Further elaborations to the system should await a fuller understanding of the more tightly defined problem.

If the embedded-coding vocoder were less idealized, a rather severe deterioration would ensue in the network performance. If, for example, the vocoder available to users was capable of synthesizing at only four rates, 2400, 4800, 8000, and 16,000 bps, then a change in the stripping priority P could result in a drastic change in the bit rate R2. For example, if users were sending at 16 kbps, and P was changed such that the lowest priority bits were now rejected, the mean bit rate R2 would immediately drop to 50% of its former value. There is a danger with such a situation of the rate oscillating at very frequent intervals between 8 kbps and 16 kbps, to achieve some intermediate rate, or, what is worse, of the

link raising the rate to 16 kbps and suddenly having to collapse to the lowest rate due to overload.

One way to alleviate this problem is to maintain several, say, 16, possible priorities, and to assign, for example, the highest priority to the 2400 bps needed to generate the low rate vocoder, the next 5 priorities to the additional 2400 bps needed for a 4800 bps vocoder, 5 priorities to the 3200 bps that bring the rate up to 8 kbps, and finally the last 5 for the 8 kbps that would allow the synthesized rate to go from 8kbps to 16 kbps. A single user would then be assigned 4 numbers for the duration of his conversation, for example, 1, 3, 10, and 15. Then the link would have 16 different priority levels to strip and each user would experience a steadier received rate.

Another alternative, which is probably more cumbersome to implement, is for each link to selectively strip a subset of the users, and to maintain a record of which users have been stripped to the lower rate. This technique would require considerable bookkeeping, and hence is probably undesirable.

Perhaps the best alternative is to invent a vocoder that is capable of synthesizing at a large variety of different rates. Work is proceeding on the development of such a vocoder, and we feel that the invested effort will be vindicated by the improvement

in network performance and simplification of network tasks that will result.

ACKNOWLEDGEMENTS

The author wishes to thank Dr. Ted Bially, Dr. Cliff Weinstein, and Dr. Ben Gold for insightful suggestions and continued interest throughout the developmental period of this simulation experiment. The author is also grateful to the above, as well as Marilyn Malpass, for critically reading early drafts of the manuscript.

REFERENCES

1. B. Gold, Proc. IEEE 65, No. 12, p. 1636-1658 (1977).

2. J. W. Forgie, "Speech Communication in Packet Switched Networks," presented at 91st Meeting of Acoustical Society of America, 5-9 April 1976.

3. C. J. Coviello and P. A. Vena, "Integration of Circuit/Packet Switching in a SENET (Slotted Envelope NETwork) Concept," National Telecommunications Conference, New Orleans, December 1975, pp. 42-12 to 42-17.

4. M. J. Fischer and T. C. Harris, IEEE Trans. Communications, Com-24, 2, pp. 195-202 (1976).

5. J. W. Forgie and A. G. Nemeth, "An Efficient Packetized Voice/Data Network using Statistical Flow Control," International Conf. on Communications, Paper #38.2, Vol. III, pp. 44-48 (1977).

6. J. W. Forgie, Journal of the Acous. Society of America, 60 Suppl. No. 1, p. s109 (Fall 1976).

7. A. G. Nemeth, "Behavior of a Link in a PVC Network," Technical Note 1976-45, Lincoln Laboratory, M.I.T., (7 December 1976), DDC AD-A036370/5.

8. K. Bullington and J. M. Fraser, Bell System Technical Journal, 38, pp. 353-364 (March 1959).

9. B. Gold, "Multiple Rate Channel Vocoding," to be presented at EASCON '78.

10. Information Processing Techniques Program Vol. II: Communications-Adaptive Internetting, Semiannual Technical Summary, Lincoln Laboratory, M.I.T., pp. 17-21 (31 March 1977), DDC A044071.

11. P. T. Brady, Bell System Technical Journal XLIV (1965).

12. L. Kleinrock, Queueing Systems, Vol. 1: Theory, (John Wiley and Sons, New York, 1975).

13. C. J. Weinstein, IEEE Trans. Communications, Com-26, 8, pp. 1253-1256 (1978).

14. P. E. Blankenship, E. M. Hofstetter, A. H. Huntoon, M. L. Malpass, S. Seneff, and V. J. Sferrino, "The Lincoln Digital Voice Terminal System," Technical Note 1975-53, Lincoln Laboratory, M.I.T. (25 August 1975), DDC AD-A017569/5.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: ESD-TR-78-274	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE *(and Subtitle)*: Computer Simulation Model for a Digital Communications Network Utilizing an Embedded Speech Encoding Technique	5. TYPE OF REPORT & PERIOD COVERED: Technical Note
6. PERFORMING ORG. REPORT NUMBER: Technical Note 1978-33	
7. AUTHOR(s): Stephanie Seneff	8. CONTRACT OR GRANT NUMBER(s): F19628-78-C-0002; ARPA Order-2006
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Lincoln Laboratory, M.I.T., P.O. Box 73, Lexington, MA 02173	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: Program Element No. 62706E, Project Code 9P10, ARPA Order 2006
11. CONTROLLING OFFICE NAME AND ADDRESS: Defense Advanced Research Projects Agency, 1400 Wilson Boulevard, Arlington, VA 22209	12. REPORT DATE: 20 October 1978 13. NUMBER OF PAGES: 66 (64 p.)
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*: Electronic Systems Division, Hanscom AFB, Bedford, MA 01731	15. SECURITY CLASS. *(of this report)*: Unclassified 15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

TN-1978-33

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

None

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

computer simulation
system
vocoder
speech signal
feedback

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

This note describes a computer simulation of a model of a rate-adaptive packetized network, based on the assumption that users have available to them an embedded-coding type vocoder. Queues are controlled at each link by stripping off lower priority packets of the encoded speech.

Conclusions are that judiciously chosen end-to-end feedback strategy can not only improve the efficiency of the system but also alleviate the problem of sudden rate drops due to link overload. The inclusion of data traffic in the model resulted in overall higher link utilization than was realized in the absence of data.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

207 650 JOB